

Please type a plus sign (+) inside this box → +

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 C.F.R. § 1.53(b))

Attorney Docket No. 500.38828X00

First Inventor or Application Identifier Tadayuki SAKAKIBARA

Title COMPUTER SYSTEM

Express Mail Label No. 

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

1. * Fee Transmittal Form (e.g., PTO/SB/17)
(Submit an original and a duplicate for fee processing)
2. Specification [Total Pages 32]
(preferred arrangement set forth below)
 - Descriptive title of the Invention
 - Cross References to Related Applications
 - Statement Regarding Fed sponsored R & D
 - Reference to Microfiche Appendix
 - Background of the Invention
 - Brief Summary of the Invention
 - Brief Description of the Drawings (if filed)
 - Detailed Description
 - Claim(s)
 - Abstract of the Disclosure
3. Drawing(s) (35 U.S.C. 113) [Total Sheets 11] 1
4. Oath or Declaration [Total Pages 5] 1
 - a. Newly executed (original or copy)
 - b. Copy from a prior application (37 C.F.R. § 1.63(d))
(for continuation/divisional with Box 16 completed)
 - i. DELETION OF INVENTOR(S)
Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. §§ 1.63(d)(2) and 1.33(b).

*** NOTE FOR ITEMS 1 & 13 IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28).**

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

5. Microfiche Computer Program (Appendix)
6. Nucleotide and/or Amino Acid Sequence Submission
(if applicable, all necessary)
 - a. Computer Readable Copy
 - b. Paper Copy (identical to computer copy)
 - c. Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

7. Assignment Papers (cover sheet & document(s))
8. 37 C.F.R. § 3.73(b) Statement Power of (when there is an assignee) Attorney
9. English Translation Document (if applicable)
10. Information Disclosure Statement (IDS)/PTO-1449 Copies of IDS Statement (IDS)/PTO-1449 Citations
11. Preliminary Amendment
12. Return Receipt Postcard (MPEP 503)
(Should be specifically itemized)
13. * Small Entity Statement(s) Statement filed in prior application (PTO/SB/09-12) Status still proper and desired
14. Certified Copy of Priority Document(s)
(if foreign priority is claimed)
15. Other: See 1 in Addendum

Continued on

16. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below and in a preliminary amendment:

Continuation Divisional Continuation-in-part (CIP) of prior application No: _____ / _____

Prior application information: Examiner _____

Group / Art Unit: _____

For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS

<input checked="" type="checkbox"/> Customer Number or Bar Code Label		020457		or <input type="checkbox"/> Correspondence address below (Insert Customer No. or Attach bar code label here)	
Name					
Address					
City		State	Zip Code		
Country		Telephone	Fax		

Name (Print/Type)	Carl L. Brundidge	Registration No. (Attorney/Agent)	29,621
Signature		Date	July 28, 2000

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231.

Attorney Docket Number: 500.38828X00

First Named Inventor or Application Identifier: Tadayuki SAKAKIBARA

Utility Patent Application Transmittal (PTO/SB/05):

15. Other Accompanying Application Parts (continued)

Credit Card Payment Form

Attachment to PTO/SB/05 (4/98) Utility Patent Application
Transmittal

1. - Figs. 1-12
 - Information Disclosure Sheet Under 37 CFR 1.56 with References

COMPUTER SYSTEM

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to a computer system, and in particular to a speculative read control scheme of cache data in a computer system having a cache between a CPU and a main memory.

Description of the Related Art

In recent years, the performance improvement of CPUs is far higher than the performance improvement of memories, and their performance difference tends to widen increasingly. A cache which operates faster than the main memory and which stores a part of contents of the main memory is used to absorb such a performance difference between the CPU and the memory and shorten the effective memory access time.

Selection of the cache capacity in a computer system largely depends upon the configuration of the computer system. In many cases, a high performance CPU itself has a cache of a large capacity. In the case where a large number of CPUs are connected, approximately 2 to 8 CPUs are connected by a bus or a switch. In the case where more CPUs are connected, they are further connected by a bus or a switch. In many cases, a hierarchical memory system is thus formed. If such a

configuration is adopted, then the access latency between the CPU and the main memory increases, and it exerts a great influence upon the performance in the case where a cache miss has occurred in a CPU. Therefore, it is

5 necessary to provide a cache in the highest class having approximately 2 to 8 CPUs connected and controlled by a bus or a switch, and thereby avoid the performance degradation when cache overflow of the CPUs has occurred.

For example, in JP-A-9-128346, a cache configuration

10 example of such a hierarchical bus system is disclosed.

The cache capacity at this time needs to be at least the total cache capacity of all CPUs connected above it. The reason can be explained as follows. When overflow has occurred in a CPU cache in the case where the above

15 described cache capacity is equal to or less than the capacity of the CPU cache, cache overflow easily occurs also in classes located below the class, resulting in a fear of remarkable degradation of the system performance.

By the way, fast devices such as SRAMs are

20 typically used in the cache in order to implement the fast access of the cache. In a typical configuration, a cache tag (tag address) and cache data are stored in the same location in this SRAM. When processing a read request, the cache tag and the cache data are

25 simultaneously read out. The cache tag is checked with a request address. In the case of a hit, the cache data can be used immediately. However, SRAMs are lower than DRAMs used in the main memory or the like in degree of

integration by at least one order. For forming a large capacity cache, it is necessary to use a large number of SRAMs. In the case where a large number of SRAMs are used, interfaces with a large number of SRAMs must be formed. Therefore, the number of pins of an LSI for controlling the cache increases, and some of the pins cannot be accommodated in one LSI. The cache tag portion is used for cache hit check. The increase of time caused by this hit check directly results in an increase of memory access latency. Therefore, an LSI having an interface with the cache tag portion needs to be accommodated in the same LSI as the CPU bus. By making an LSI having an interface with the cache data portion different from the LSI including the CPU bus and providing the interface with a data width nearly equal to the CPU bus width, the pin neck of LSIs can be eliminated.

On the other hand, as a scheme for improving the hit factor of the cache, there is a set associative scheme. For example, in JP-A-5-225053, there are disclosed a scheme of conducting tag comparison of a set associative cache and its speed increase.

In hit check in the set associative scheme, cache tags of a plurality of cache lines of a plurality of ways are read out, and hit check is conducted simultaneously in a plurality of lines. At this time, it remains to be seen which data of a plurality of lines is used until the cache hit check is completed. In a cache (on chip cache) mounted on a CPU, it is typical to adopt

such a scheme that cache access latency is reduced by conducting readout of the cache data simultaneously with readout of the cache tag and selecting only necessary data after the cache hit check has been completed.

5 FIG. 12 shows an example of a configuration of such a set associative cache. FIG. 12 shows a 4-way set associative cache including N entries.

Each entry includes four ways, a 0th way 1000, a first way 1001, a second way 1002, and a third way 1003.

10 Information contained in the cache includes STAT 1004 indicating the state (valid or invalid) of the cache, a cache tag (address) 1005, and cache data 1006. In a typically adopted method, a low order address of a memory address is used as the entry number of the cache, and a 15 high order address is used as the cache tag. In an on-chip cache, a cache tag 1005 and cache data 1006 are stored together as shown in FIG. 12. Therefore, it is possible to read simultaneously the cache tag 1005 and the cache data of each of the ways 1000 to 1003 of a 20 pertinent entry, and immediately select data by using a way number subjected to cache hit.

If it is attempted to implement a set associative cache having a large capacity, however, it is necessary to separate an LSI having an interface with the 25 cache data from an LSI having interfaces with the CPU bus and the cache tag. In this case, the cache tag and the cache data cannot be read at the same time. Therefore, the cache tag and the cache data are read out separately.

If at this time the data width between the LSI having the interface with the cache tag and the LSI having the interface with the cache data is only approximately the CPU bus due to a physical restriction, then it takes a
5 too long time to read out all cache data of a plurality of lines into the LSI of the CPU bus side. For example, in the case where the CPU bus width is 8 bytes and the line size is 32 bytes, it takes $4 \text{ cycles} \times 4 \text{ ways} = 16$ cycles to transfer lines corresponding to 4 ways from the
10 cache data side LSI to the cache tag side LSI. This means that it takes 16 cycles whenever the cache is referred to. As a result, the performance is remarkably degraded. For preventing this performance degradation, it becomes necessary to read out cache data after a
15 result of cache hit check is found. However, this causes an increase of access latency of the cache. Related art is disclosed in JP-A-9-128346 and JP-A-5-225053, for example.

SUMMARY OF THE INVENTION

20 In the case where a large capacity cache of the set associative scheme or the like is provided between the CPU and the main memory as described above, it becomes necessary to put the cache tag portion and the cache data portion in separate LSIs and manage them under
25 the restrictions of, for example, the number of pins of LSIs. In the case where such a configuration is adopted, there is a problem that the cache readout latency

increases if the cache tag is read out and the cache hit check is conducted, and thereafter the cache data is read out.

An object of the present invention is to
5 realize shortening of the cache data readout time in the case where the cache tag portion and the cache data portion are managed in separate LSIs as described above, in a computer system having a cache such as an n way set associative cache located in a class between the CPU and
10 the main memory in hierarchy.

In order to achieve the above described object, in accordance with an aspect of the present invention, an advanced or speculative read request is issued to a controller of the cache data portion before conducting
15 the cache hit check. Thus data supplied from the cache is read in advance and held in the controller. In the case where a cache hit has occurred, the read request based on the cache hit check is issued to the controller to read the data subjected to advanced speculative
20 readout is read out.

BRIEF DESCRIPTION OF THE DRAWING

FIG. 1 is a block diagram showing a computer system of an embodiment of the present invention;

FIG. 2 is a diagram showing relations among an
25 address supplied from a CPU, a cache tag, and a cache entry number;

FIG. 3 is a diagram showing a configuration

example of a cache tag section;

FIG. 4 is a diagram showing a configuration example of a cache data section;

5 FIG. 5 is a processing flow diagram of a coherent controller according to the first embodiment of the present invention;

FIG. 6 is a detailed block diagram of a cache data controller;

10 FIG. 7 is a diagram showing a configuration example of a speculative read request buffer in the cache data controller;

FIG. 8 is a diagram showing a speculative read data buffer in the cache data controller;

15 FIG. 9 is a processing flow diagram of a request controller in the cache data controller according to the first embodiment of the present invention;

FIG. 10 is a processing flow diagram of a coherent controller according to a second embodiment of the present invention;

20 FIG. 11 is a processing flow diagram of a request controller in a cache data controller according to a second embodiment of the present invention; and

FIG. 12 is a diagram showing a configuration example of a conventional 4-way set associative cache.

25 DESCRIPTION OF THE EMBODIMENTS

Hereafter, embodiments of the present invention will be described in detail by referring to drawing.

FIG. 1 shows a computer system of an embodiment of the present invention. The present system includes two CPUs, i.e., CPU(0) 1 and CPU(1) 2, a storage controller (SCU) 4, a cache tag section 5, a cache data controller 6, a cache data section 7, a main memory 8, and a bus 3 for connecting the CPU(0) 1, CPU(1) 2, and the SCU 4. Furthermore, the SCU 4 includes a bus 16, a memory access request queue 17, a write data buffer 18, a read response data buffer 19, a coherent controller 20, a memory access controller 21, a bus 22, and buses 23 to 29 for connecting them. Here, the number of CPUs (i.e., the number of nodes) is two. As a matter of course, however, the number of nodes may be two or more, or the number may be one.

Although not illustrated, it is assumed that each of the CPU(0) 1 and CPU(1) 2 has a built-in cache. Furthermore, it is assumed that each of the cache tag section 5 and the cache data section 7 includes an SRAM which is a high speed memory device. It is assumed that the main memory 8 includes a DRAM which is a low speed memory device. Furthermore, it is assumed that the cache of the present system formed of the cache tag section 5 and the cache data section 7 is a 4-way set associative cache.

FIG. 2 shows relations among a request address supplied from the CPU(0) 1 and CPU(1) 2, a cache tag, and a cache entry number. In the present embodiment, it is assumed that the request address supplied from the CPU(0)

1 and CPU(1) 2 has 32 bits and the number of cache entries is 256 K entries. Furthermore, it is assumed that the cache line size is 32 bytes. In FIG. 2, numeral 100 denotes the request address (ADR <31:0>) output from 5 the CPU(0) 1 and CPU(1) 2. Since the cache line size is 32 bytes, six low-order bits of the ADR 100 indicate an address in the cache line. Since the number of cache entries is 256 K, 18 bits of ADR <24:7> become a cache entry number 102. ADR <31:25> which is the remaining 10 high-order address becomes a cache tag 101.

FIGS. 3 and 4 show configuration examples of the cache tag section 5 and the cache data section 7. As shown in FIG. 3, the cache tag section 5 is a 256 K entry, 4-way aggregate of cache states (STATs) 210 and cache 15 tags 211. Furthermore, as shown in FIG. 4, the cache data section 7 is a 256 K entry, 4-way aggregate of cache data 220. There is a one-to-one correspondence between entries and ways of the cache tag section 5 and the cache data section 7. For example, a cache tag corresponding 20 to cache data stored in a block of a 0th entry and a 0th way of the cache data section 7 is stored in a block of a 0th entry and a 0th way of the cache tag section 5. The cache state (STAT) 210 indicates whether cache data (cache line) of the pertinent block is valid or invalid.

25 Returning back to FIG. 1, the cache data controller 6 accepts a cache data read/write request issued by the SCU 4, and reads/writes cache data to/from the cache data section 7. A path 12 is a path for

sending an access request fed from the SCU 4 to the cache data controller 6. A path 13 is a path for exchanging data between the SCU 4 and the cache data controller 6.

5 A path 30 is a signal line to be used by the cache data controller 6 to conduct read/write control on the cache data section 7. A path 31 is a path for exchanging data between the cache data controller 6 and the cache data section 7. In the present embodiment, the number of signals of the paths 30 and 31 is large because of the 4-

10 way set associative cache. Therefore, it is physically impossible to provide pins in the SCU 4 directly for the cache data section 7. Accordingly, the cache data controller 6 is formed as a chip separate from the chip of the SCU 4. As a result, the paths 12 and 13 are

15 smaller than the paths 30 and 31 in the number of signals.

The memory access request queue 17 in the SCU 4 is a queue for buffering memory access requests issued by the CPU(0) 1 and CPU(1) 2 and sending a memory access request to the coherent controller 20 if the coherent controller 20 is not busy. The data buffer 18 is a buffer for temporarily storing write data supplied from the CPU(0) 1 and CPU(1) 2. The data buffer 19 is a buffer for temporarily storing read response data to be returned to the CPU(0) 1 and CPU(1) 2. The coherent controller 20 determines whether a memory access request issued by the CPU(0) 1 and CPU(1) 2 conducts a cache hit, and issues an access request to the cache data controller 6 or the memory access controller 21. The memory access

controller 21 effects access control of the main memory 8 in accordance with the access request issued by the coherent controller 20.

The coherent controller 20 resolves the request
5 address 100 supplied from the CPU(0) 1 and CPU(1) 2 as shown in FIG. 2, reads out cache tags 211 from each way of a pertinent entry of the cache tag section 5 shown in FIG. 3 by using the cache entry number 102, compares them with the cache tag 101 of the request address 100, and
10 thereby conducts cache hit check. This cache hit check itself is basically the same as that of the conventional technique.

Operation of an embodiment in the computer system of FIG. 1 will now be described.

15 In the case where data required for execution of an instruction is not stored in the built-in cache, the CPU(0) 1 or CPU(1) 2 issues a memory access request to the SCU 4 via the bus 3. In the SCU 4, the memory access request is stored in the memory access request queue 17 via the bus 16. In the case of a write request, data is also sent from the CPU(0) 1 or CPU(1) 2. In the SCU 4, therefore, write data is stored in the data buffer 18 via the bus 16. If the coherent controller 20 is not busy, a memory access request is sent from the memory access request queue 17 to the coherent controller 20.
20
25

By referring to the cache tag section 5, the coherent controller 20 determines whether the received memory access request hits the cache. However, if, in

the case of a read request, a data is read out from the data cache 7 via the cache data controller based on a result of the cache hit decision is obtained, the access latency becomes large. Then, before conducting a cache 5 hit decision by referring to the cache tag section 5, the coherent controller 20 issues a request for conducting advanced or speculative readout to the cache data controller 6 (when the readout request is received from the processor bus). The speculative (SP) readout request 10 may be formatted to include an address area, a read/write ID area and as SP bit area indicating whether the request is speculative. Otherwise the request may be formatted by only an entry address, if it is speculative. Thus, the coherent controller 20 reads out data which should be 15 read out when a hit has occurred, from the cache data section 7 into the cache data controller 6 in advance. When a hit has occurred, the coherent controller 20 uses this data read in advance.

FIG. 5 is a processing flow of an embodiment of 20 the coherent controller 20. Hereafter, detailed operation of the coherent controller 20 will be described by referring to FIG. 5.

Upon accepting a memory access request from the CPU(0) 1 or CPU(1) 2 (step 300), the coherent controller 25 20 determines whether the request is a read request (step 301). If the request is a read request, the coherent controller 20 issues an advanced or speculative read request to the cache data controller 6 via the paths 25

and 12 (step 302). At the same time, the coherent controller 20 sends a cache entry number of the pertinent read request to the cache tag section 5 via the path 23 and a path 10, and reads out cache tags corresponding to 5 4 ways of the pertinent entry from the cache tag section 5 via a path 11 and the path 24 (step 303). The coherent controller 20 determines whether the cache tags read out from the cache tag section 5 hit the cache tag of the read request (step 304). When a hit has occurred, the 10 coherent controller 20 issues a read request to the cache data controller 5 via the paths 25 and 12 (step 305). The read request at this time includes a way number for which the hit has occurred, along with a cache entry number. In the case where a cache miss has occurred, the 15 coherent controller 20 issues a read request to the memory access controller 21 (step 306), and newly registers a cache tag of the pertinent memory access request in a desired way of the pertinent entry of the cache tag section 5 via the paths 23 and 10 (step 307). 20 The memory access controller 21 accesses the main memory 8 via the path 27 and a path 14, and reads out data onto a path 15 and the path 28. If response data is returned from the main memory 8, the coherent controller 20 issues a write request to the cache data controller 6 in order 25 to register this response data with the cache data section 7, and sends the response data to the cache data controller 6 via the bus 22 and the paths 26 and 13 as write data (step 108). At the same time, the coherent

controller 20 stores the response data in the data buffer 19 from the bus 22 in order to send the response data to the CPU (step 309).

- If the request received from the CPU(0) 1 or 5 CPU(1) 2 is a write request, then the coherent controller 20 reads out cache tags corresponding to 4 ways of the pertinent entry from the cache tag section 5 in the same way as the case of the read request (step 310), and determines whether there has occurred a cache hit (step 10 311). If there has occurred a cache hit, the coherent controller 20 issues a write request to the cache data controller 6 via the paths 25 and 12 (step 312). At the same time, the coherent controller 20 sends write data to the cache data controller 6 via the paths 26 and 13 (step 15 313). The write request at this time includes a way number for which the hit has occurred, along with a cache entry number. In the case where a cache miss has occurred, the coherent controller 20 issues a write request to the memory access controller 21 (step 314). 20 At the same time, the coherent controller 20 sends write data to the main memory 8 via the paths 28 and 15 (step 315). The memory access controller 21 accesses the main memory 8 via the paths 27 and 14, and writes the data into the main memory 8.
- Especially in the case where a read request has been accepted, the coherent controller 20 thus has a function of issuing an advanced or speculative read request to the cache data controller 6 before conducting 25

a cache hit check by using the cache tag section 5. In the case of a write request, the operation is basically the same as the operation of the conventional technique.

The configuration and operation of the cache
5 data controller 6 will now be described. With reference to FIG. 1, the cache data controller 6 exchanges data with the cache data section 7 in accordance with an advanced or speculative read request, a read request, and a write request supplied from the coherent controller 20
10 via the path 12.

FIG. 6 is a detailed block diagram of the cache data controller 6. The cache data controller 6 includes a request controller 400, a speculative read request buffer 401, an address comparator section 402,
15 speculative read data buffers 403 to 406, buses 407 and 408 to 411, selectors 412 and 413 to 416, and paths 417 to 428.

The request controller 400 decodes a request received from the coherent controller 20 via the path 12,
20 determines processing to be conducted in the cache data controller 6 on the basis of a kind of the accepted request, and controls respective components. The speculative read request buffer 401 is a buffer for holding a speculative read request received from the
25 coherent controller 20. The speculative read data buffers 403 to 406 are buffers for holding data read out from the cache data section 7 in accordance with a speculative read request. As shown in FIG. 4, the cache

data section 7 of the present embodiment is 4-way set associative. Data of the 0th way are stored in the speculative read data buffer 403. Data of the first way are stored in the speculative read data buffer 404. Data 5 of the second way are stored in the speculative read data buffer 405. Data of the third way are stored in the speculative read data buffer 406. The address comparator section 402 determines whether an advanced or speculative read request, a read request, or a write request has the 10 same cache entry as a request stored in the speculative read request buffer 401. See Figs. 2 and 3.

FIGS. 7 and 8 show configuration examples of the speculative read request buffer 401 and the speculative read data buffers 403 to 406. As shown in 15 FIG. 7, the speculative read request buffer 401 includes a plurality of entries. Each entry includes a valid bit (V) 500 and a cache entry number 501. The valid bit 500 is a bit indicating that the entry is valid or invalid. The cache entry number 501 is a cache entry number which 20 is the subject of a speculative read request stored in the pertinent entry. As shown in FIG. 8, each of the speculative read data buffers 403 to 406 also includes a plurality of entries. Cache data (32 bytes) 600 read out from the cache data section 7 speculatively by a 25 speculative read request is stored in each entry.

There is one-to-one correspondence between entries of the speculative read request buffer 401 and entries of the speculative read data buffers 403 to 406.

For example, if it is assumed that a cache entry number of a certain speculative read request is stored in the 0th entry of the speculative read request buffer 401, cache data corresponding to 4 ways read out from the 5 cache data section 7 speculatively by the speculative read request are stored in the 0th entry of the speculative read data buffers 403 to 406. The number m of entries of the speculative read request buffer 401 and the speculative read data buffers 403 to 406 may be an 10 arbitrary number. Furthermore, the buffers 401 and 403 to 406 may be formed as one body.

FIG. 9 is a processing flow of the request controller 400 in an embodiment. Hereafter, detailed operation of the cache data controller 6 will be 15 described centering around the request controller 400 by referring to FIG. 9.

Upon receiving a request from the coherent controller 20 via the paths 12 and 417 (step 700), the request controller 400 first determines whether the 20 request is a speculative read request (step 701). If the request is a speculative read request, then the request controller 400 determines whether a request to the same cache entry is stored in the speculative read request buffer 401 beforehand (step 702). To be concrete, the 25 request controller 400 outputs the cache entry number of the speculative read request to the path 419. In addition, the request controller 400 reads out cache entry numbers of respective entries of the speculative

read request buffer 401, makes the address comparator section compare the cache entry number of the speculative read request with the cache entry numbers of respective entries, receives results of the comparison via the path 5 420, and thereby determines whether the same cache entry as that of the speculative read request is stored in the speculative read request buffer 401 beforehand. If the same cache entry is stored, the newly received speculative read request is discarded. If a request to 10 the same cache entry is not stored in the speculative read request buffer 401, then the request controller 400 determines whether the speculative read request buffer 401 is full (step 703). If the speculative read request buffer 401 is not full, then the request controller 400 15 registers a new speculative read request with an empty entry in the speculative read request buffer 401 via the path 428 (step 705). If the speculative read request buffer 401 is full, then the request controller 400 invalidates the oldest entry in the speculative read 20 request buffer 401 (step 704), and thereafter registers a new request. By the way, such an invalidation algorithm is well known as a LRU (Least Recently Used) method. Detailed description thereof will be omitted. The registered speculative read request is transferred to the 25 cache data section 7 via the paths 418 and 30 as a read request. Cache data corresponding to 4 ways are read out from the pertinent cache entry of the cache data section 7 (step 706). The cache data are newly stored in an

entry of the speculative read data buffers 403 to 406, corresponding to the entry in the speculative read request buffer 401 with which the speculative read request has been registered via the path 31, the buses 5 408 to 411, and the paths 423 to 426 (step 707).

As a result, in the case where the speculative read request buffer 401 is full, new cache data is overwritten and stored in the pertinent entry of the speculative read data buffers 403 to 406, corresponding 10 to the invalid entry in the speculative read request buffer 401.

If the request received from the coherent controller 20 is not a speculative read request, but a read request (step 708), then the request controller 400 15 checks whether an address (cache entry number) of the same cache entry as that of the read request is stored in the speculative read request buffer 401 beforehand (step 709). How to check is the same as that in the case of the speculative read request. If there is the same cache 20 entry, then the request controller 400 reads out data from the pertinent entry of the speculative read data buffers 403 to 406, and sends the data to the path 13 via the selectors 413 to 416, the selector 412, and the bus 407 as response data (step 710). In other words, the 25 request controller 400 outputs a selection signal of the speculative read request buffer side on the path 422, and outputs a hit way number included in the read request to the path 421 as a selection signal. As a result, data

corresponding to 4 ways read out from the pertinent entry of the speculative read data buffers 403 to 406 are first selected by the selectors 413 to 416. Subsequently, data corresponding to the hit way number in the pertinent 4 ways is selected by the selector 412, and sent to the path 13 via the bus 407 as response data. Thereafter, the pertinent entry of the speculative read request buffer 401 is invalidated (step 711).

If there is not an address of the same cache entry as that of the read request in the speculative read request buffer 401, then the request controller 400 transfers the pertinent read request to the cache data section 7 via the paths 418 and 30, selects cache data corresponding to 4 ways read out from the pertinent cache entry of the cache data section 7 by using the selectors 413 to 416 via the buses 408 to 411, selects data corresponding to the hit way number included in the cache data by using the selector 412, and sends out the selected data from the bus 407 to the path 13 as response data (step 712).

This case occurs when the data read from the cache data section 7 into the speculative read data buffers 403 to 406 in advance by the speculative read request is invalidated by a write request (preceding write request) hereafter described before a subsequent corresponding read request.

In the case whether the request received from the coherent controller 20 via the paths 12 and 417 is

neither a speculative read request nor a read request,
i.e., also in the case where the request is a write
request, the request controller 400 determines whether an
address to the same cache entry is stored in the
5 speculative read request buffer 401 beforehand (step 713).

If the address is present, the request
controller 400 invalidates the pertinent entry of the
speculative read request buffer 401 (step 714).

Subsequently, the request controller 400 sends out a
10 write request to the cache data section 7 via the paths
418 and 30. At the same time, the request controller 400
sends out cache data received from the coherent
controller 20 via the path 13 to the path 31 via the bus
407, the path 427, and the buses 408 to 411, and writes
15 the data into a specified way number of a specified entry
of the cache data section 7 (step 715).

In the case where a request to the same entry
as the write request received from the coherent
controller 20 is present in the speculative read request
20 buffer 401, the pertinent entry is invalidated at the
step 714 in FIG. 9. The reason why doing so is that
otherwise the data in the cache data section 7 is
rewritten by the write operation and noncoincidence with
data in the speculative read data buffers 403 to 406
25 occurs. By virtue of the invalidation processing of the
step 714, rewritten new data is read out from the cache
data section 7 at step 712 in a subsequent read request
for the same cache entry.

In the case where a read request received from the coherent controller 20 is a request to the same entry as a request in the speculative read request buffer 401, the cache data controller 6 selectively returns data read 5 in advance and stored in the speculative read data buffers 403 to 406, instead of data supplied from the cache data section 7, in the present embodiment as shown in FIG. 9. As a result, access latency of the cache data section 7 can be reduced. If the coherent controller 20 10 issues a speculative read request while conducting the cache hit check as shown in FIG. 5, therefore, it becomes possible to reduce cycles corresponding to the cache hit check time from the memory access latency.

FIGS. 10 and 11 show processing flows of the 15 coherent controller 20 and the request controller 400 in the cache data controller 6 in another embodiment of the present invention.

FIG. 10 is the processing flow of the coherent controller 20. FIG. 10 is different from FIG. 5 in that 20 a step 800 has been added. In the case where the speculative read request issued to the cache data controller 6 at the step 302 results in a cache miss, a request (speculative read data discarding request) for invalidating the speculative read data read in advance by 25 the pertinent speculative read request is issued to the cache data controller 6 at the step 800. As a result, the cache data controller 6 can invalidate unused speculative read data stored in the speculative read data

buffers 403 to 406. Accordingly, effective use of the speculative read request buffer 401 and the speculative read data buffers 403 to 406 becomes possible.

FIG. 11 is a processing flow of the request controller 400 included in the cache data controller 6. FIG. 11 is different from FIG. 9 in that steps 900 and 901 have been added. The steps 900 and 901 are a processing flow conducted in the case where a speculative read cancellation request has been accepted from the coherent controller 20. In other words, upon receiving a speculative read data discarding request from the coherent controller 20 (step 900), the request controller 400 invalidates an entry in the speculative read request buffer 401 in which a cache entry number of a speculative read request corresponding to the pertinent speculative read data cancellation request has been registered (step 901). As a result, effective use of the speculative read request buffer 401 and the speculative read data buffers 403 to 406 becomes possible. If each of the buffers 401 and 403 to 406 is formed with a margin of a certain degree in the number of entries, it becomes possible to eliminate the full state and it also becomes possible to make the full control itself of the steps 703 and 704 unnecessary.

In the case where a read request received from the coherent controller 20 is a request to the same entry as a request in the speculative read request buffer 401, the cache data controller 6 reads out data from some of

the speculative read data buffers 403 to 406, instead of data supplied from the cache data section 7, in the present embodiment as well in the same way as the the above described first embodiment. As a result, access 5 latency of the cache data section 7 can be reduced. If the coherent controller 20 issues a speculative read request while conducting the cache hit check, therefore, it becomes possible to reduce cycles corresponding to the cache hit check time from the memory access latency.

10 Heretofore, in the embodiments of the present invention, it has been assumed that the cache is a 4 way set associative. However, the number of ways may be an arbitrary number of at least one. Furthermore, it is a matter of course that the present invention is not 15 limited to a set associative cache, but the present invention can be widely applied to a computer system using such a cache scheme that the cache tag portion and the cache data portion are managed in separate LSIs.

WHAT IS CLAIMED IS:

1. A computer system including a CPU, a memory, and a cache located in a hierarchy class between said CPU and said memory, said computer system comprising:

a coherent controller for determining whether a request supplied from said CPU hits said cache to thereby issue a request to said cache or said memory; and

a cache data controller for controlling reading or writing of data registered in said cache, in accordance with a request issued by said coherent controller;

wherein upon accepting a read request from said CPU, said coherent controller conducts hit decision of said cache, issues an advanced speculative read request to said cache data controller, and issues a read request to said cache data controller if the hit decision is a cache hit.

2. A computer system according to claim 1, wherein said cache data controller comprises:

means responsive to acceptance of an advanced speculative read request issued by said coherent controller, for reading data from said cache and holding the data; and

means responsive to acceptance of a read request and a cache hit decision for sending said held speculative read data to said CPU as response data.

3. A computer system according to claim 1, wherein if a hit decision is a cache miss, said

coherent controller issues a speculative read data discarding request to said cache data controller, and wherein upon accepting a speculative read data discarding request issued by said coherent controller, said cache data controller cancels speculative read data of a speculative read request corresponding to the speculative read data discarding request.

4. A computer system according to claim 1, wherein said cache is an n-way associative cache.

5. A computer system according to claim 4, wherein said cache data controller accepts an advanced speculative read request corresponding to n ways issued by said coherent controller, reads out data corresponding to n ways from said cache, and hold the data.

6. A cache data control method in a computer system including a CPU, a memory, and a cache located in a hierarchy class between said CPU and said memory, said cache data control method comprising the steps of:

receiving a request from said CPU;
determining whether said request is an advanced speculative data request;

if said request is a speculative data request,
determining whether a request to the same cache entry as said advanced speculative data request is stored in a speculative read request buffer beforehand; and

if the request is stored in a speculative read request buffer beforehand, disregarding the advanced speculative data request received from said CPU.

7. A cache data control method according to claim 6, wherein if, as a result of determining whether a request to the same cache entry as said advanced speculative data request is stored in a speculative read request buffer beforehand, said advanced speculative data request is not stored in said speculative read request buffer, said method further comprises the steps of:

determining whether said speculative data request buffer is full;

if said speculative data request buffer is not full, registering said advanced speculative data request with an empty entry in said speculative data request buffer;

if said speculative data request buffer is full, invalidating an oldest entry included in said speculative data request buffer; and

registering said advanced speculative read request with said speculative data request buffer.

8. A cache data control method according to claim 6, wherein if, as a result of determining whether said request received from said CPU is an advanced speculative data request, said request is not an advanced speculative data request, but a read request, said method further comprises the steps of:

determining whether an address of the same cache entry as said read request is stored in said speculative read request buffer;

if the address of the same cache entry as said

advanced read request is stored in said speculative read request buffer, reading out data from a pertinent entry of said speculative read request buffer; and
transmitting said data as response data.

9. A cache data control method according to claim 8, wherein if, as a result of determining whether an address of the same cache entry as said read request is stored in said speculative read request buffer, the address is not stored, said method further comprises the steps of:

transferring said read request to said cache;
selecting cache data read out from a pertinent cache entry of said cache; and

transmitting said cache data as response data.
10. A cache data control method according to claim 6, wherein if, as a result of determining whether said request received from said CPU is an advanced speculative data request, said request is not an advanced speculative data request, but a write request, said method further comprises the steps of:

determining whether an address of the same cache entry as said write request is stored in said speculative read request buffer;

if the address of the same cache entry as said write request is stored in said speculative read request buffer, invalidating a pertinent entry of said speculative request buffer;

transmitting said write request to a data

section of said cache; and

 writes said write request into a specified entry of said cache data section.

11. A cache data control method in a computer system including a CPU, a memory, and a cache located in a hierarchy class between said CPU and said memory, said cache data control method comprising the steps of:

 receiving a memory access request from said CPU;

 determining whether said memory access request is a read request;

 if said memory access request is a read request, issuing an advanced speculative read request to a cache data controller, and sending a cache entry number of said read request to a cache tag section;

 reading out a cache tag of said cache entry number from said cache tag section;

 determining whether said cache tag read out hits a cache tag of said read request; and

 upon a hit, issuing a read request to said cache controller.

12. A cache data control method according to claim 11, wherein if a cache miss occurs between the cache tag read out from said cache tag section and the cache tag of said read request, said cache data control method further comprises the steps of:

 issuing a read request; and

 registering the cache tag of said memory access

request with said cache tag section.

13. A cache data control method according to claim 11, wherein if the request received from the CPU is a write request, said cache data control method further comprises the steps of:

reading a cache tag from said cache tag section and determining whether a cache hit has occurred;

if as a result of said determination a cache hit has occurred, issuing a write request to said cache data controller; and

sending write data from a data buffer to said cache data controller.

14. A cache data control method according to claim 13, wherein if the determination on the cache tag read out from said cache tag section results in a cache miss, said cache data control method further comprises the steps of:

issuing a write request; and

sending write data from the data buffer to said memory.

15. A computer system according claim 1, wherein said cache is a set associative cache.

16. A cache data control method according to claim 6, wherein a set associative cache is used as said cache.

17. A cache data control method in a computer system including a CPU, a storage controller, a cache tag section connected to said storage controller, a cache data section, and a cache data controller connected

between said cache data section and said storage controller, said cache data control method comprising the steps of:

reading cache data from said cache data section to hold in said cache data controller in response to a first read request from said storage controller; and

sending said cache data from said cache data controller to said storage controller in response to a second read request issued from said storage controller based on cache hit result from said cache tag section.

18. A cache data control method according to claim 17, further comprising a step of including a bit indicating whether a request issued from said storage controller is either one of said first and second data requests.

ABSTRACT OF THE DISCLOSURE

A cache data control system and method for a computer system in which in a memory read processing, a coherent controller issues an advanced speculative read request for (speculatively) reading data from a cache data section in advance to a cache data controller, before reading a cache tag from a cache tag section and conducting cache hit check. When a cache hit has occurred, the cache data controller returns the data subjected to speculative reading as response data at the time when the cache data controller has received a read request issued by the coherent controller.

FIG. I

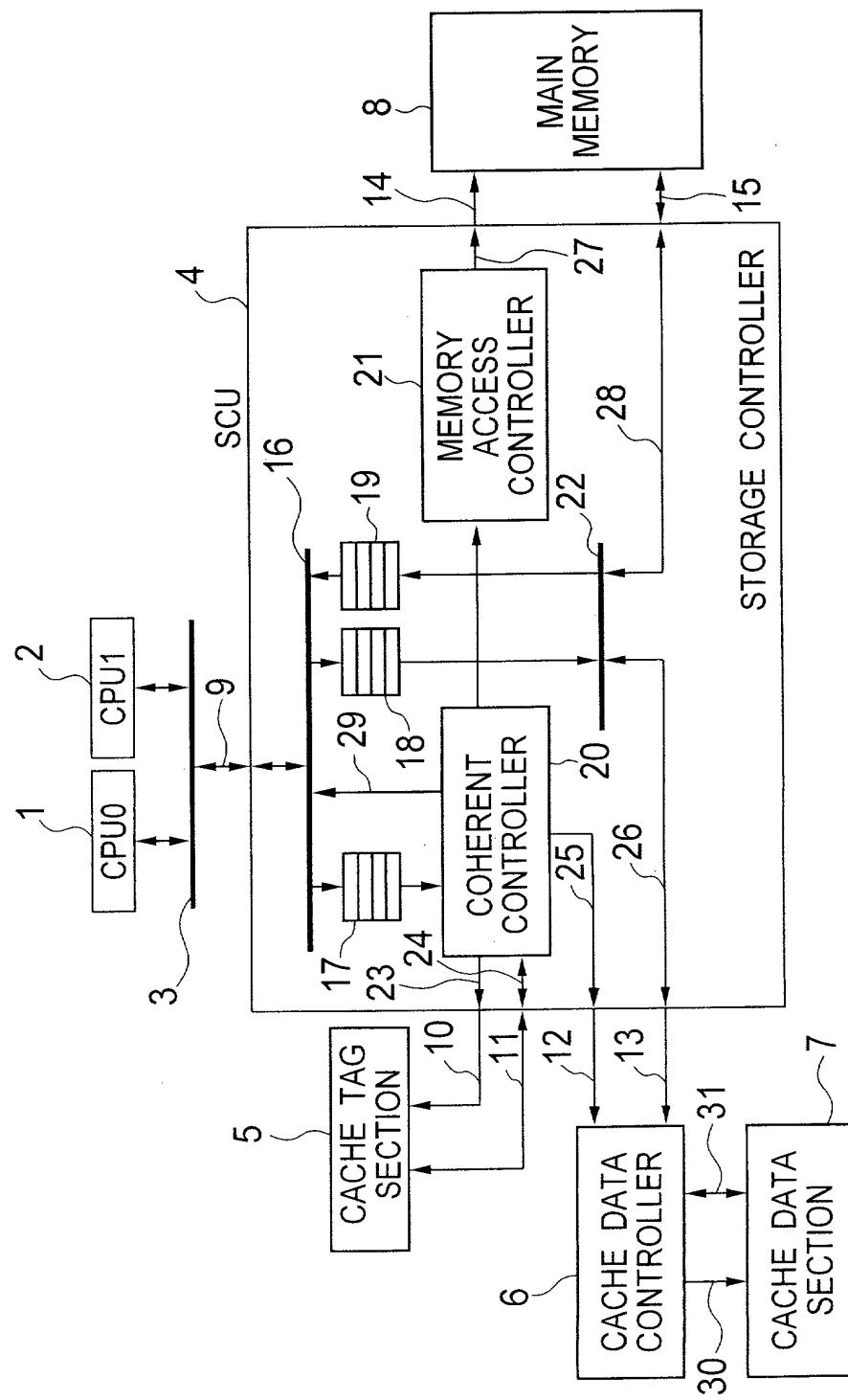
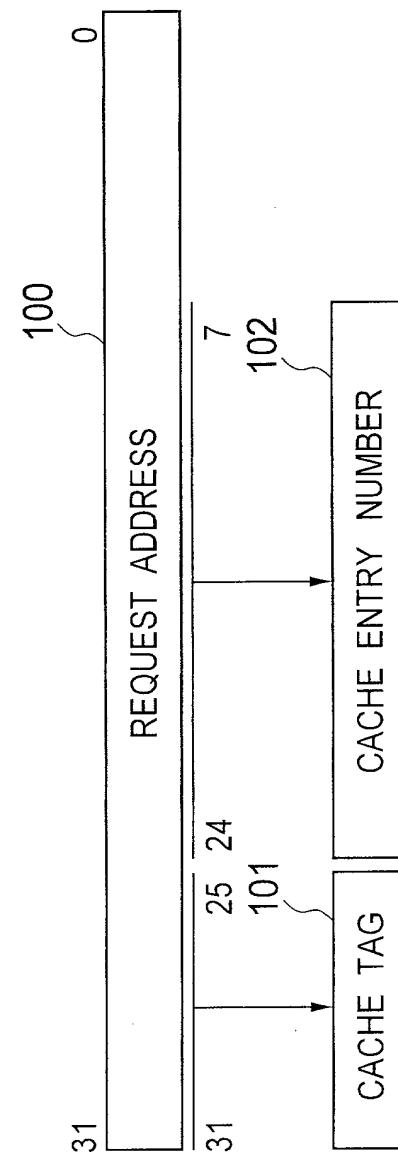


FIG. 2



F | G. 3

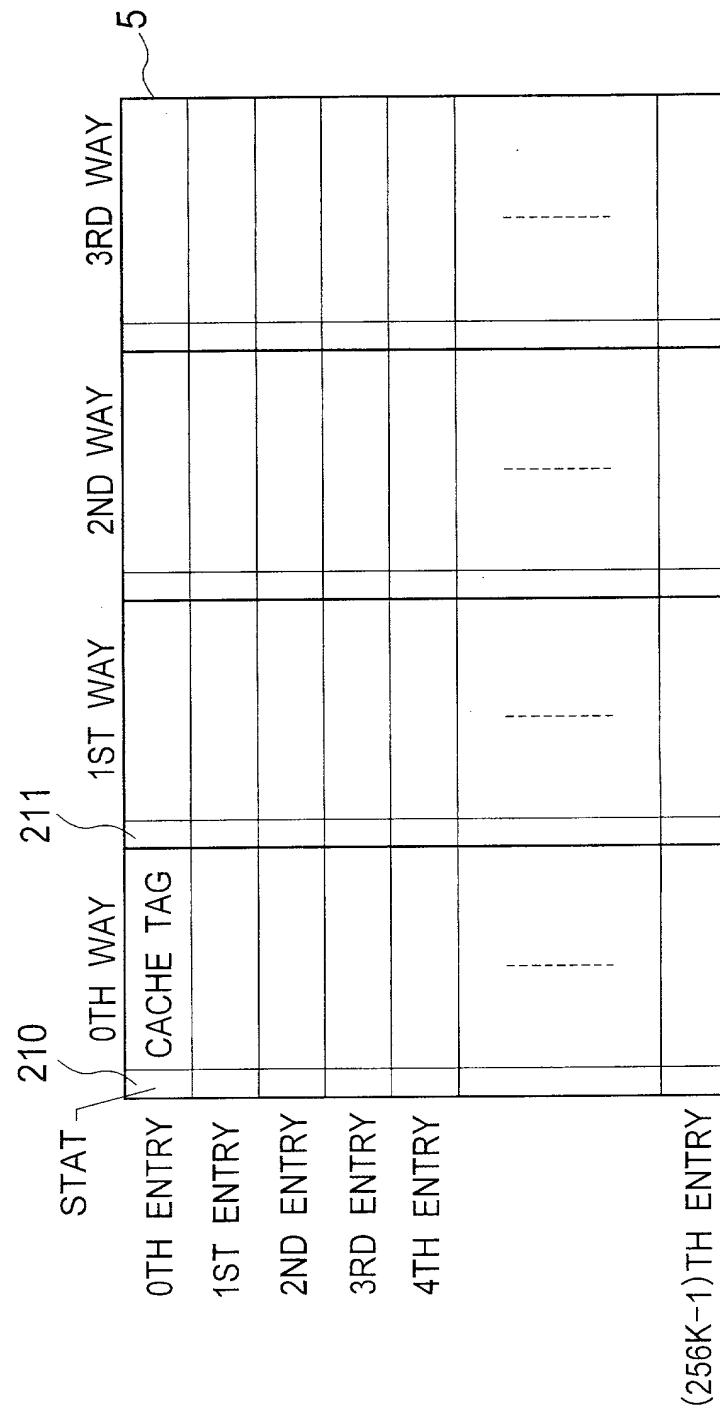
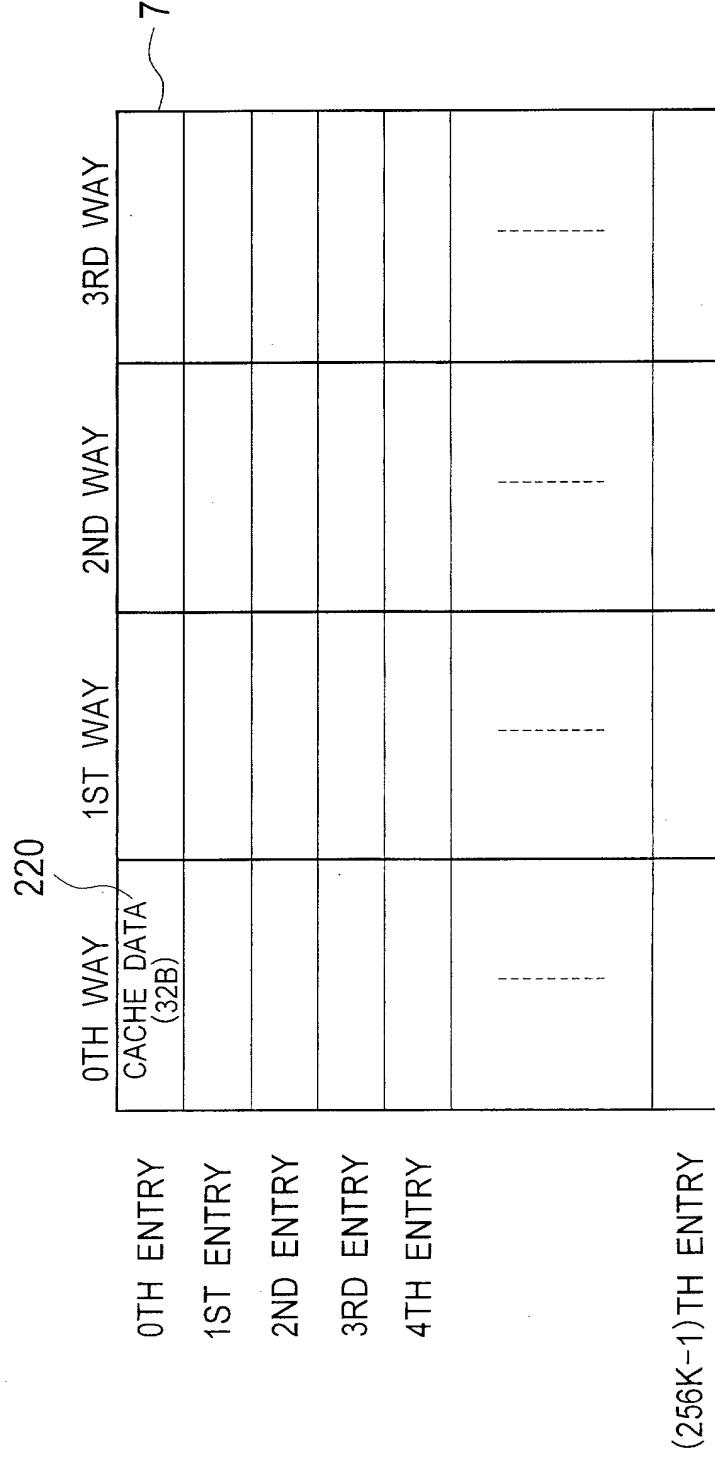


FIG. 4



50

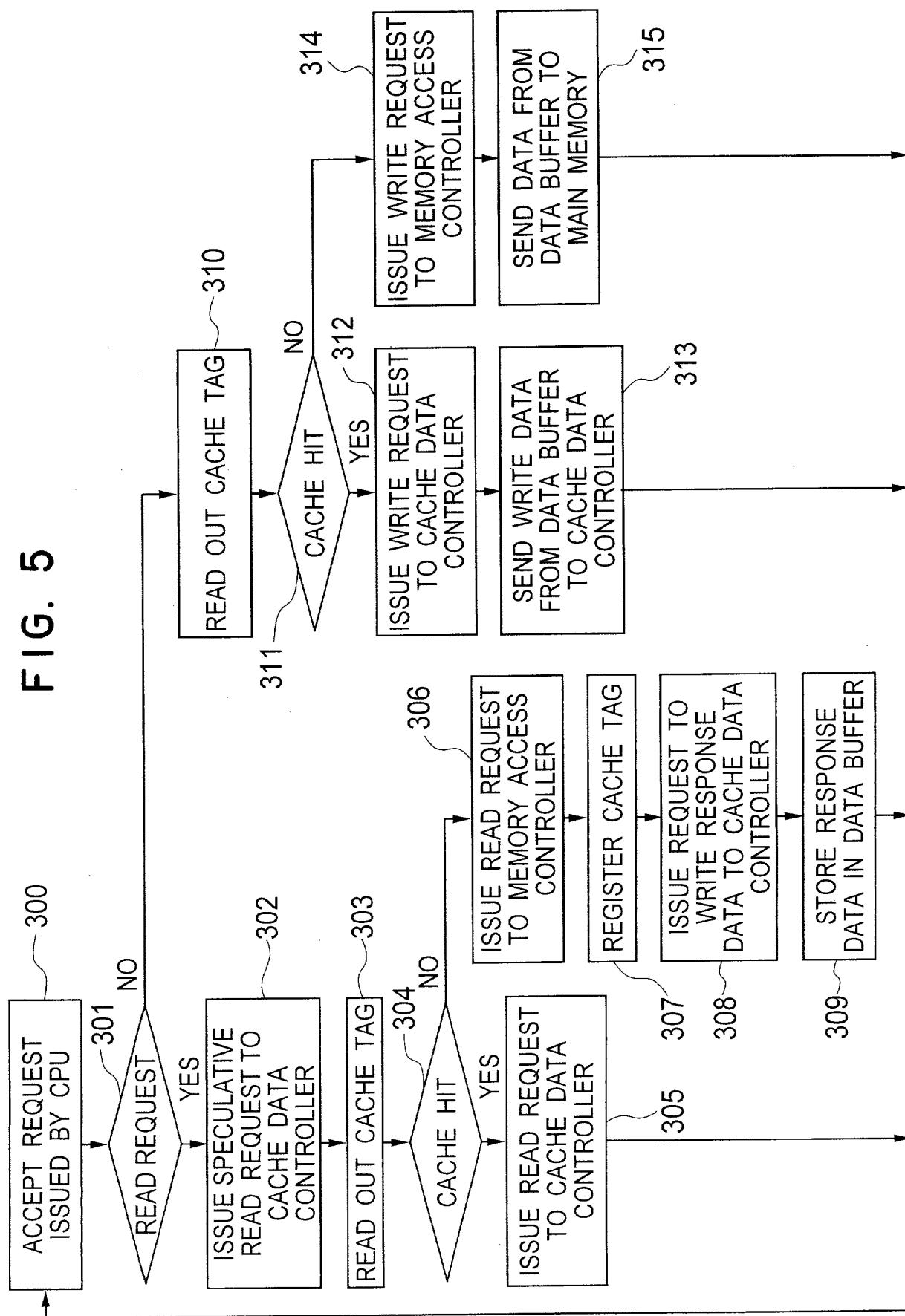


FIG. 6

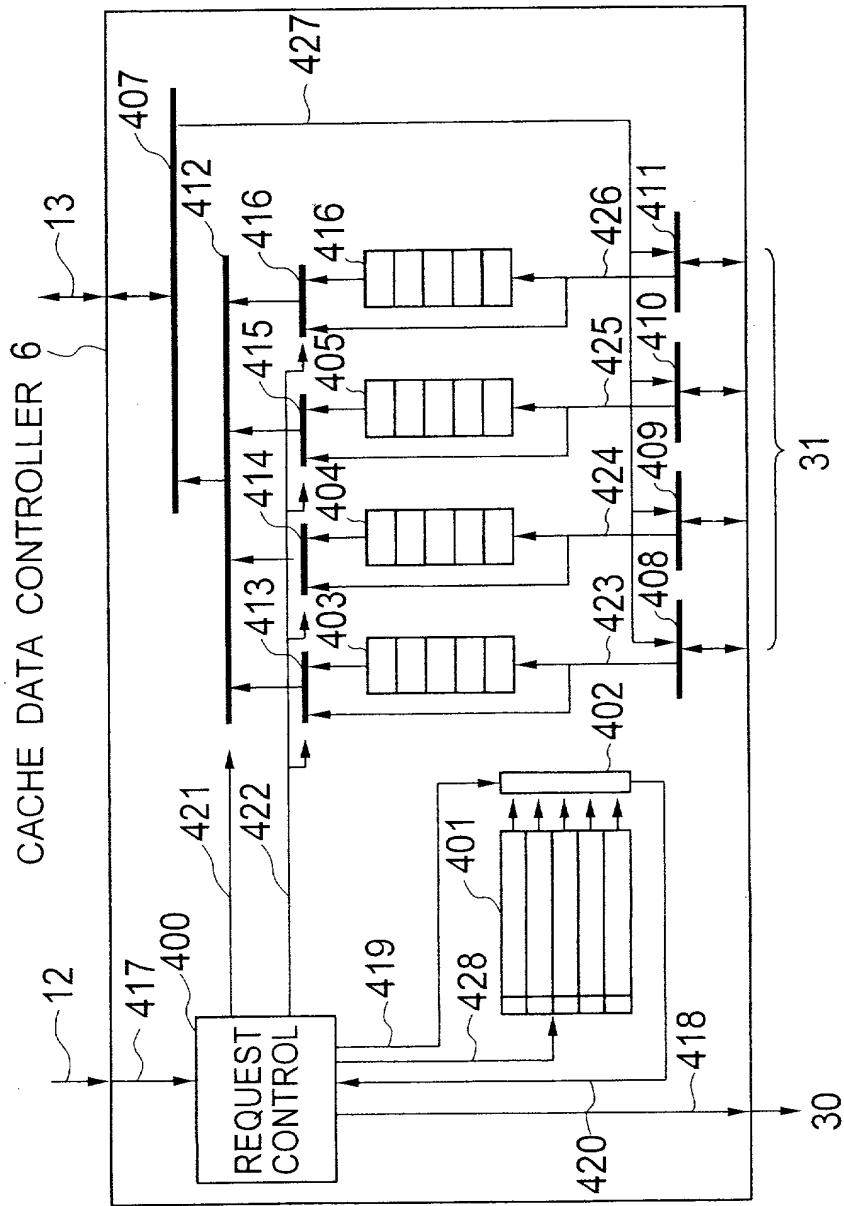


FIG. 7

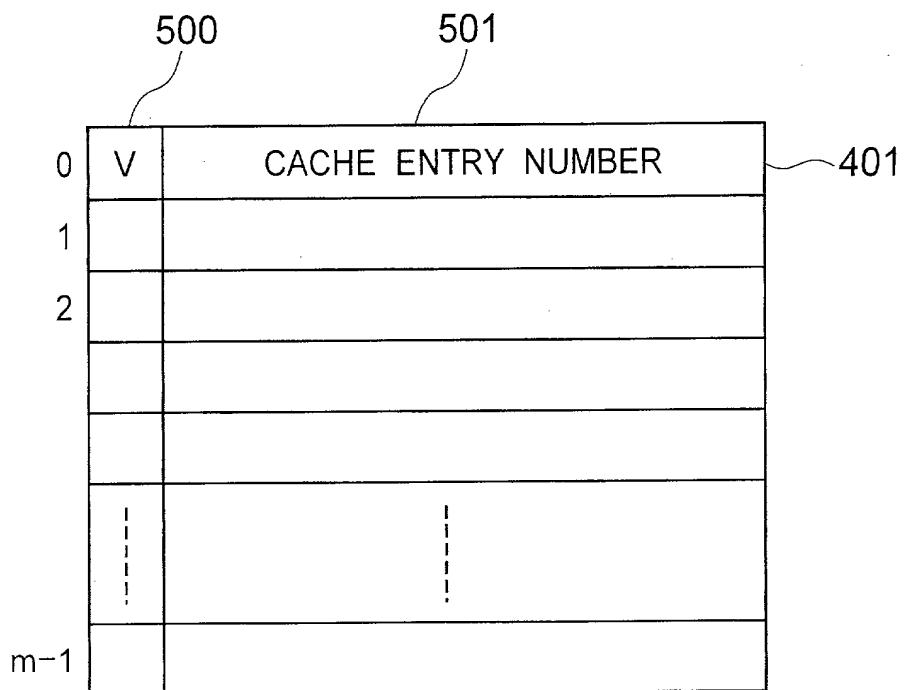


FIG. 8

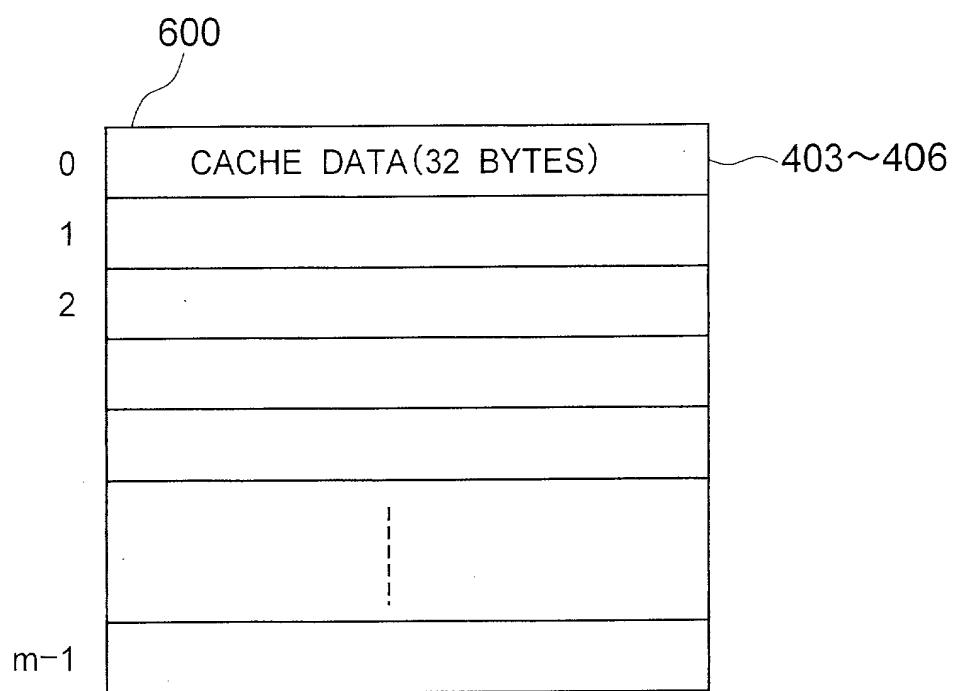


FIG. 9

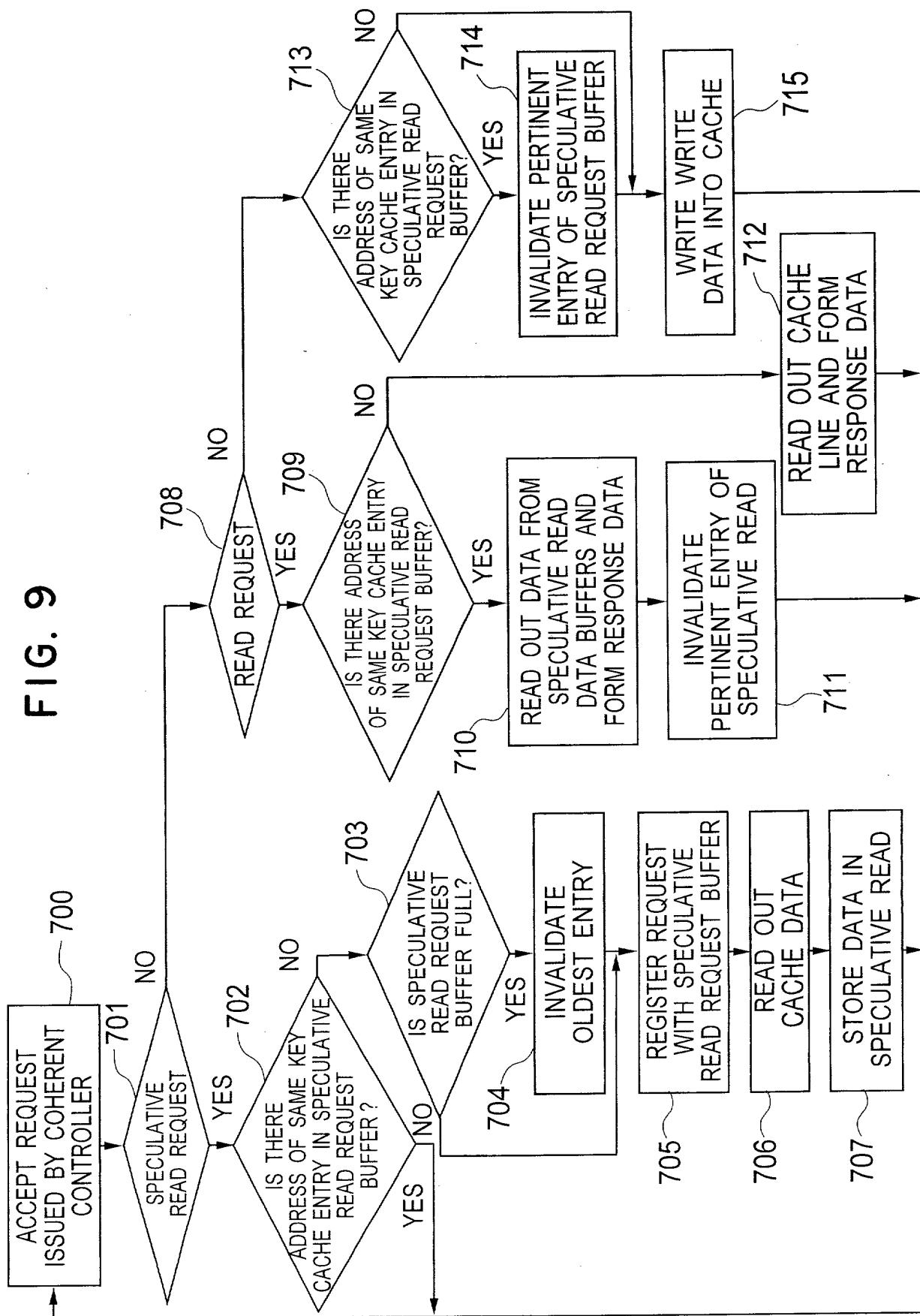


FIG. 10

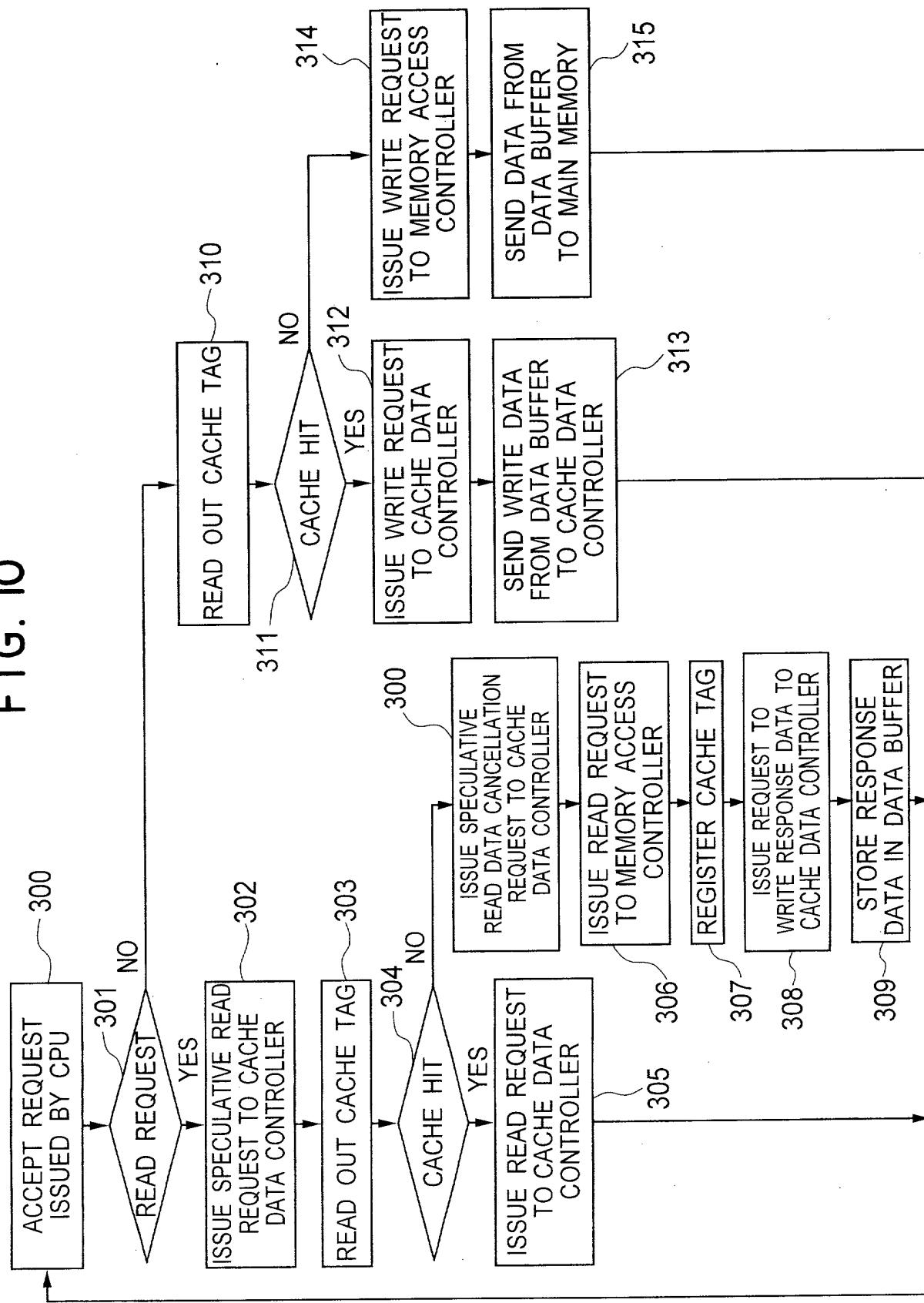


FIG. 11

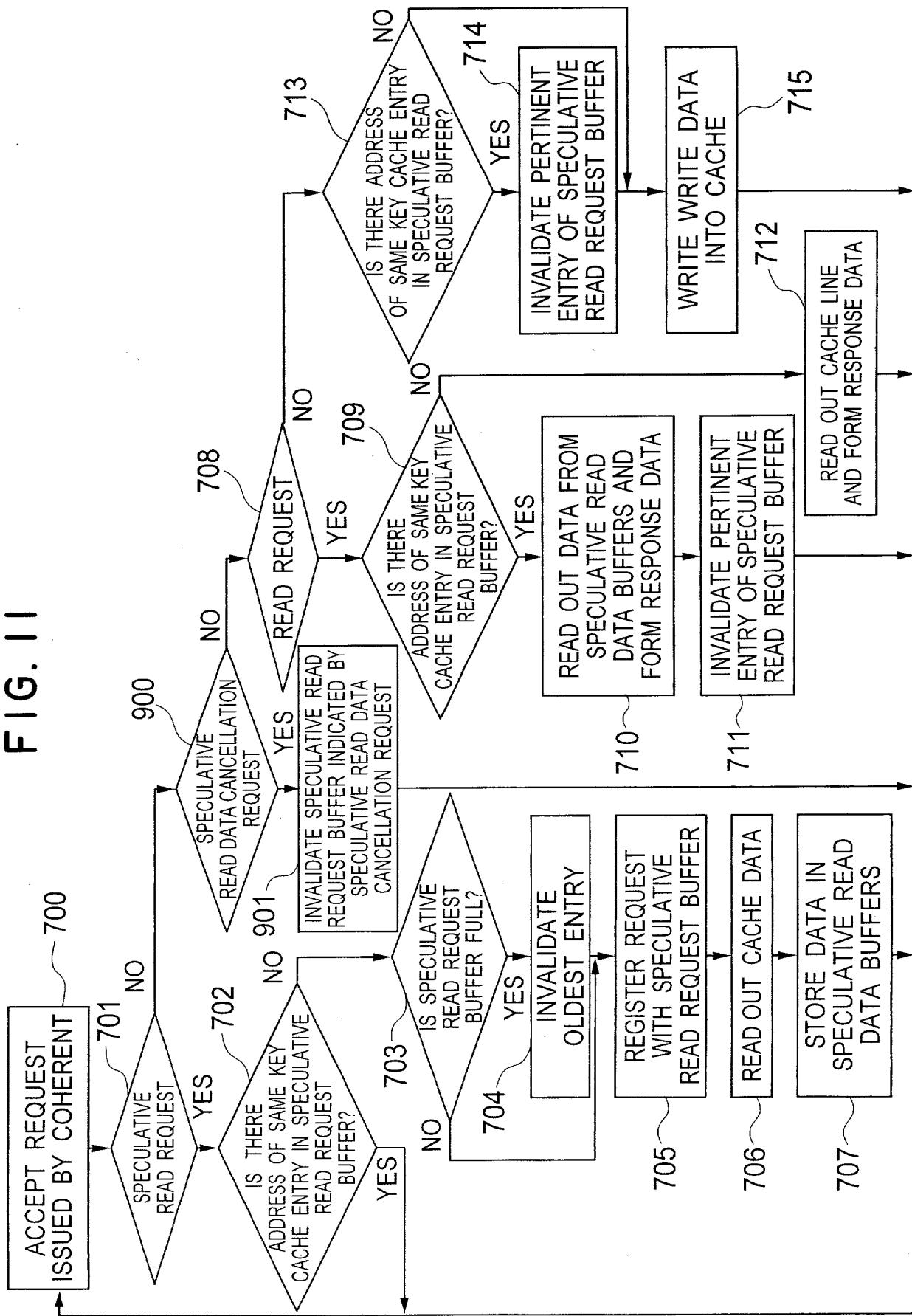
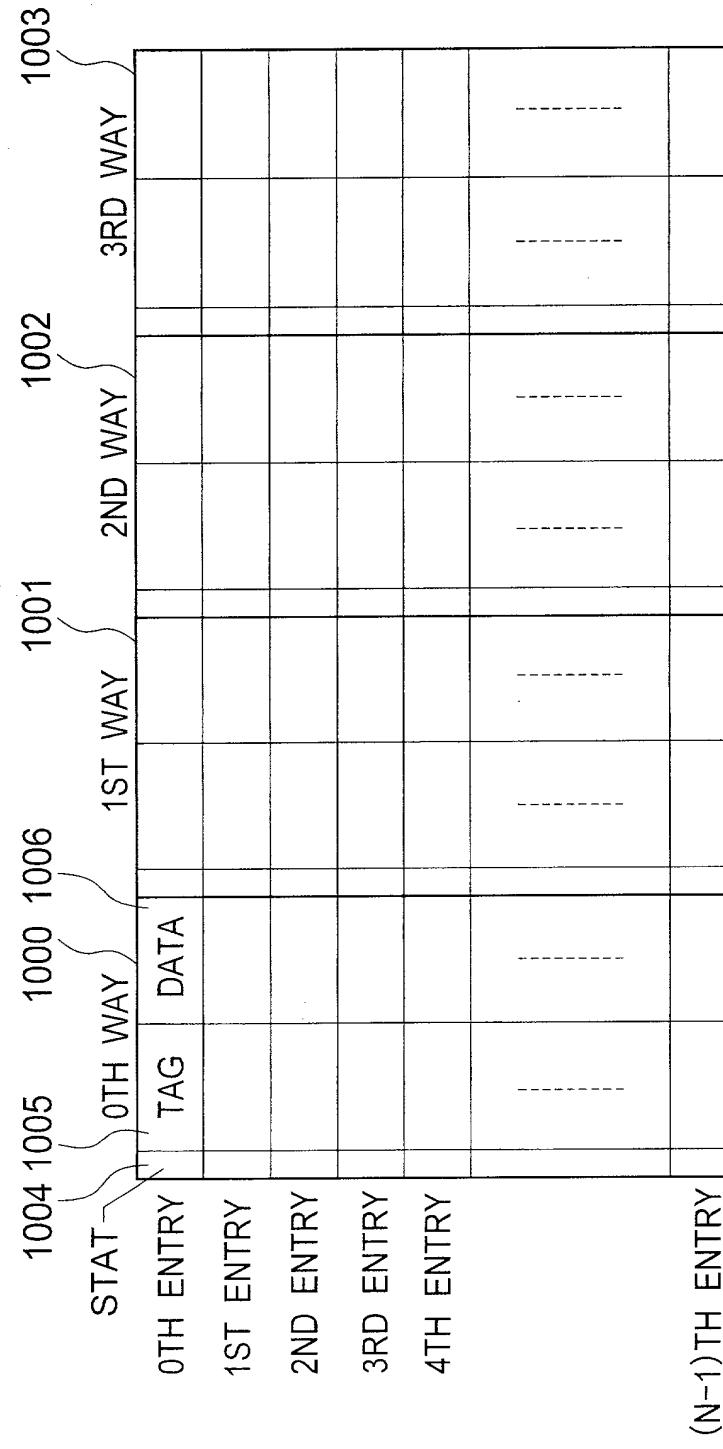


FIG. 12



349905233

PTO/SB/106(8-96)

Approved for use through 9/30/98. OMB 0651-0032

Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Declaration and Power of Attorney For Patent Application

特許出願宣言書及び委任状

Japanese Language Declaration

日本語宣言書

下記の氏名の発明者として、私は以下の通り宣言します。

As a below named inventor, I hereby declare that:

私の住所、私書箱、国籍は下記の私の氏名の後に記載された通りです。

My residence, post office address and citizenship are as stated next to my name.

下記の名称の発明に関して請求範囲に記載され、特許出願している発明内容について、私が最初かつ唯一の発明者（下記の氏名が一つの場合）もしくは最初かつ共同発明者であると（下記の名称が複数の場合）信じています。

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

COMPUTER SYSTEM

上記発明の明細書（下記の欄で×印がついていない場合は、本書に添付）は、

The specification of which is attached hereto unless the following box is checked:

___月___日に提出され、米国出願番号または特許協定条約国際出願番号を_____とし、
(該当する場合) _____に訂正されました。

was filed on
as United States Application Number or
PCT International Application Number
_____ and was amended on
_____ (if applicable).

私は、特許請求範囲を含む上記訂正後の明細書を検討し、内容を理解していることをここに表明します。

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

私は、連邦規則法典第37編第1条56項に定義されるとおり、特許資格の有無について重要な情報を開示する義務があることを認めます。

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Japanese Language Declaration (日本語宣言書)

私は、米国法典第35編119条(a) - (d)項又は365条(b)項に基き下記の、米国以外の國の少なくとも一ヵ国を指定している特許協力条約365(a)項に基づく国際出願、又は外国での特許出願もしくは発明者証の出願についての外国優先権をここに主張するとともに、優先権を主張している、本出願の前に出願された特許または発明者証の外国出願を以下に、枠内をマークすることで、示している。

Prior Foreign Application(s)

外国での先行出願

11-216614	Japan
(Number) (番号)	(Country) (国名)
(Number) (番号)	(Country) (国名)

私は、第35編米国法典119条(e)項に基いて下記の米国特許出願規定に記載された権利をここに主張いたします。

(Application No.) (出願番号)	(Filing Date) (出願日)

私は、下記の米国法典第35編120条に基いて下記の米国特許出願に記載された権利、又は米国を指定している特許協力条約365条(c)に基づく権利をここに主張します。また、本出願の各請求範囲の内容が米国法典第35編112条第1項又は特許協力条約で規定された方法で先行する米国特許出願に開示されていない限り、その先行米国出願書提出日以降で本出願書の日本国内または特許協力条約国提出日までの期間中に入手された、連邦規則法典第37編1条56項で定義された特許資格の有無に関する重要な情報について開示義務があることを認識しています。

(Application No.) (出願番号)	(Filing Date) (出願日)

私は、私自身の知識に基づいて本宣言書中で私が行なう表明が真実であり、かつ私の入手した情報と私の信じるところに基づく表明が全て真実であると信じていること、さらに故意になされた虚偽の表明及びそれと同等の行為は米国法典第18編第1001条に基づき、罰金または拘禁、もしくはその両方により処罰されること、そしてそのような故意による虚偽の声明を行なえば、出願した、又は既に許可された特許の有効性が失われることを認識し、よってここに上記のごとく宣誓を致します。

I hereby claim foreign priority under Title 35, United States Code, Section 119 (a)-(d) or 365(b) of any foreign application(s) for patent or inventor's certificate, or 365(a) of any PCT international application which designated at least one country other than the United States, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or PCT International application having a filing date before that of the application on which priority is claimed.

Priority Not Claimed
優先権主張なし

30/July/1999
(Day/Month/Year Filed) (出願年月日)
(Day/Month/Year Filed) (出願年月日)

I hereby claim the benefit under Title 35, United States Code, Section 119(e) of any United States provisional application(s) listed below.

(Application No.) (出願番号)	(Filing Date) (出願日)

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s), or 365(c) of any PCT international application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of application.

(Status: Patented, Pending, Abandoned) (現況: 特許許可済、係属中、放棄済)
(Status: Patented, Pending, Abandoned) (現況: 特許許可済、係属中、放棄済)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Japanese Language Declaration (日本語宣言書)

委任状： 私は下記の発明者として、本出願に関する一切の手続きを米特許商標局に対して遂行する弁理士または代理人として、下記の者を指名いたします。（弁護士、または代理人の氏名及び登録番号を明記のこと）

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith (list name and registration number)

Donald R. Antonelli, Reg. No. 20,296; David T. Terry, Reg. No. 20,178; Melvin Kraus, Reg. No. 22,466; William I. Solomon, Reg. No. 28,565; Gregory E. Montone, Reg. No. 28,141; Ronald J. Shore, Reg. No. 28,577; Donald E. Stout, Reg. No. 26,422; Alan E. Schiavelli, Reg. No. 32,087; James N. Dresser, Reg. No. 22,973 and Carl I. Brundidge, Reg. No. 29,621

書類送付先

Send Correspondence to:

Antonelli, Terry, Stout & Kraus, LLP
Suite 1800
1300 North Seventeenth Street
Arlington, Virginia 22209

直接電話連絡先：（氏名及び電話番号）

Direct Telephone Calls to: (name and telephone number)

Telephone: (703) 312-6600
Fax: (703) 312-6666

唯一または第一発明者	Full name of sole or first inventor Tadayuki SAKAKIBARA	
発明者の署名	日付	Inventor's signature
		Tadayuki Sakakibara
住所	Date	
	7/13/00	
国籍	Residence Kamakura, Japan	
私書箱	Citizenship Japan	
	Post Office Address c/o Hitachi, Ltd., Intellectual Property Group New Marunouchi Bldg. 5-1, Marunouchi 1-chome, Chiyoda-ku, Tokyo 100-8220, Japan	

（第二以降の共同発明者についても同様に記載し、署名すること）
(Supply similar information and signature for second and subsequent joint inventors.)

第二共同発明者	Full name of second joint inventor, if any Isao OHARA		
第二共同発明者の署名	日付	Second inventor's signature	Date <i>Isao Ohara</i> 7/13/00
住所	Residence Hadano, Japan		
国籍	Citizenship Japan		
私書箱	Post Office Address c/o Hitachi, Ltd., Intellectual Property Group New Marunouchi Bldg. 5-1, Marunouchi 1-chome, Chiyoda-ku, Tokyo 100-8220, Japan		
第三共同発明者	Full name of third joint inventor, if any Hideya AKASHI		
第三共同発明者の署名	日付	Third inventor's signature	Date <i>Hideya Akashi</i> 7/13/00
住所	Residence Kunitachi, Japan		
国籍	Citizenship Japan		
私書箱	Post Office Address c/o Hitachi, Ltd., Intellectual Property Group New Marunouchi Bldg. 5-1, Marunouchi 1-chome, Chiyoda-ku, Tokyo 100-8220, Japan		
第四共同発明者	Full name of fourth joint inventor, if any Yuji TSUSHIMA		
第四共同発明者の署名	日付	Fourth inventor's signature	Date <i>Yuji Tsushima</i> 2000. July. 13
住所	Residence Hadano, Japan		
国籍	Citizenship Japan		
私書箱	Post Office Address c/o Hitachi, Ltd., Intellectual Property Group New Marunouchi Bldg. 5-1, Marunouchi 1-chome, Chiyoda-ku, Tokyo 100-8220, Japan		
第五共同発明者	Full name of fifth joint inventor, if any Satoshi MURAOKA		
第五共同発明者の署名	日付	Fifth inventor's signature	Date <i>Satoshi Muraoka</i> 7/13/00
住所	Residence Yokohama, Japan		
国籍	Citizenship Japan		
私書箱	Post Office Address c/o Hitachi, Ltd., Intellectual Property Group New Marunouchi Bldg. 5-1, Marunouchi 1-chome, Chiyoda-ku, Tokyo 100-8220, Japan		

(第六以降の共同発明者についても同様に記載し、署名すること)

(Supply similar information and signature for sixth and subsequent joint inventors.)

第六共同発明者		Full name of sixth joint inventor, if any	
第六共同発明者の署名	日付	Sixth inventor's signature	Date
住所		Residence	
国籍		Citizenship	
私書箱		Post Office Address	
第七共同発明者		Full name of seventh joint inventor, if any	
第七共同発明者の署名	日付	Seventh inventor's signature	Date
住所		Residence	
国籍		Citizenship	
私書箱		Post Office Address	
第八共同発明者		Full name of eighth joint inventor, if any	
第八共同発明者の署名	日付	Eighth inventor's signature	Date
住所		Residence	
国籍		Citizenship	
私書箱		Post Office Address	
第九共同発明者		Full name of ninth joint inventor, if any	
第九共同発明者の署名	日付	Ninth inventor's signature	Date
住所		Residence	
国籍		Citizenship	
私書箱		Post Office Address	

(第十以降の共同発明者についても同様に記載し、署名をすること)

(Supply similar information and signature for tenth and subsequent joint inventors.)